JSON：数据格式

1. JSON格式

- 所谓的JSON格式就是对象类型；
- JSON，全称是 JavaScript Object Notation，是 JavaScript对象标记法。
- JSON是一种轻量级、基于文本的、可读的格式。
- JSON 的名称中虽然带有JavaScript，但这是指其语法规则是参考JavaScript对象的，而不是指只能用于 JavaScript 语言。
- 前后端进行交互的时候的唯一的可选的（用夸张的描述来表示其重要性）数据格式；

2. JSON 的语法规则

- 对象是一个无序的“ ‘名称/值’ 对”集合。一个对象以 {左括号 开始， }右括号 结束。每个“名称”后跟一个 :冒号 ； “ ‘名称/值’ 对”之间使用 ,逗号 分隔。
- 数组（Array）用方括号(“[]”)表示。数组是值（value）的有序集合。一个数组以 [左中括号 开始，]右中括号 结束。值之间使用 ,逗号 分隔。
- 对象（0bject）用大括号(“{}”)表示。
- 名称/值对(name/value）组合成数组和对象。
- 名称(name）置于双引号中，值（value）可以是双引号括起来的字符串（string）、数值(number)、true、false、 null、对象（object）或者数组（array）。这些结构可以嵌套。
- 字符串（**string**）是由双引号包围的任意数量Unicode字符的集合，使用反斜线转义。一个字符（character）即一个单独的字符串（character string）。
- 并列的数据之间用逗号(“,”）分隔，最后一个“名称/值对“之后不要加逗号

```
<script>
        /* JSON格式的数据非常常用，非常重要*/
        var json = {
            code:1,
            msg:'登录成功'
        };
        console.log(typeof json, json);

        var students = ['A', 'B'];
        console.log(students);
        /**
            标准的前后端数据通信的JSON格式的数据
        */
        var data = {  //对象用“{}”表示
            code:1,  /* 1 的含义是我们自己规定的 */
```

```
        message: '数据获取成功',  //并列的数据之间用“,”隔开
        data:[ //数组用“[]”表示
            {
                sid:1,//并列的数据之间用“,”隔开
                name:'C',
                gender:2,
                school:'六盘水师范学院'
            },//并列的数据之间用“,”隔开
            {
                sid:23,
                name:'D',
                gender:2,
                schoole:'贵阳大学'
            },
            {
                sid:269,
                name:'E',
                gender:1,
                schoole:'昆明理工',
                score:[
                    {
                        course:'math',
                        score:95
                    },
                    {
                        course:'english',
                        score:102
                    }
                ]
            }
        ]
    };
    console.log(data.data[0].school);
    console.log(`
                ${data.data[2].name}
                    的
                ${data.data[2].score[1].course}
                    成绩是
                ${data.data[2].score[1].score}
            `);
```

```
    </script>
```

```
object {code: 1, msg: '登录成功'}
    code: 1
    msg: "登录成功"
    [[Prototype]]: Object
(2) ['A', 'B']
    0: "A"
    1: "B"
    length: 2
    [[Prototype]]: Array(0)
六盘水师范学院

    E
     的
    english
     成绩是
     102
```

函数：function

- 函数也是一个对象，可以封装一些功能（代码），在需要的时候调用执行
- 使用function声明函数
- 函数名的命名规范和变量一致（本质上变量名和函数名一样的，都是标识符）；
- 函数名后面跟上小括号，里面放形参，当然也可以为空，表示在调用的时候不需要传递实参；相当于在函数内部声明了对应的变量
- 小括号后面跟大括号，大括号里面放的是函数体(功能模块需要很多行js代码，就用{}括起来)；

```
function 函数名(形参1, 形参2...形参N) {   //可以定义任意个参数，用逗号分开
    语句...       //函数体
    return;       //返回函数执行结果
}
// 调用函数
函数名(); //通过调用函数名来执行函数体代码
```

```
// 声明函数
function sum(num1,num2){
    console.log(num1+num2);  //函数体
}

```

```
// 调用函数
sum(1,3) //4
sum(6, 5) //11
```

- 在函数体里面使用return返回函数执行结果；
- 如果没有返回值，那么函数的返回值就是undefined；如果函数中不写return，也会返回underfined；
- return 后面的代码不会执行；

```
function add(a, b){
    ...    //函数体
    return a + b; // return 后的代码不执行
    console.log('我不会被执行，因为前面有 return');
}
var num = add(21,6); // 调用函数，传入两个实参，并通过 num 接收函数返回值
console.log(num);    // 27
```

- return只能返回一个值，如果有逗号隔开的多个值，会返回最后一个值；

```
function add(a, b){
    ...    //函数体
    return a,b;
}
var num = add(21,6); // 调用函数，传入两个实参，并通过 resNum 接收函数返回值
alert(num);          // 6
```

- 函数名后面跟上小括号表示函数的执行（函数的调用），在执行的时候会在内部产生一个this，参考下面对this的说明；
- 在调用函数的时候给形参传的值是 实参；

形参和实参统称为参数:

参数	说明
形参	**形**式上的**参**数，**函数定义**的时候， 传递的参数，当前并不知道是什么
实参	**实**际上的**参**数，**函数调用**的时候， 传递的参数，实参是传递给形参的

对象里面的函数我们称之为方法

在函数（方法）执行的时候会产生一个this；

- This是当前函数（方法）执行的上下文，就是执行场景；
- 顶层对象是window，是个全局变量；
- This：谁调用这个函数，this就指向谁（暂时这么理解，不精准）；

1. 通过函数名()直接调用的：this指向window，函数执行中this指向的并不是函数本身，（函数也是对象），而是调用它的对象，我们知道，浏览器的全局变量是window，所以这里的this指向window，当然window.mynmme 为 undifined

```
function obj(){
      var name = "ABC";
      console.log(this);
      console.log(this.name);
  }
  obj();//等价于 window.obj();
```

2. 通过对象.函数名()调用的：this指向这个对象；直接调用obj();实际上是window.obj(),所以调用obj的是全局window，所以这里的this指向window， 而将obj赋值给a对象的fun属性后，调用a.fun(),就是a来调用obj（），所以this指向a;

```
var myname = "我是window的name";
 function obj(){
      var myname = "ABC";
      console.log(this);
      console.log(this.myname);
  }
  var a = {
      fun:obj, //将函数obj 赋值给a.fun
      myname:"我是a的name"
  };
  obj();
  a.fun();
```

3. 函数作为数组的一个元素，通过数组下标调用的：this指向这个数组

```
  var myname = "我是window的name";
  function obj(){
      var myname = "ABC";
      console.log(this);
      console.log(this.myname);
```

```
    console.log(this[0].myname);//打印数组第0个元素myname属性
}
var myarr = [{myname:"我是myarr[0]的name"},obj];//新建以数组对象
myarr.myname = "我是myarr的name";//为数组对象添加myname属性
myarr[1]();//数组下标方式调用函数
```

4. 函数作为window内置函数的回调函数调用时：this指向window.setTimeout(func,XXms);setInterval(func,XXms)等
5. 函数作为构造函数，用new关键字调用时：this指向的是new出的新对象
6. 通过函数指定，用apply()\call()\bind() 方法指定this
7. 箭头函数中的this问题，指向为定义时的this，而不是执行时的this

```
<script>
        /* var 声明的全局变量会追加到顶层对象window身上*/
        var a = 100, b = 200;
        var obj = {
            a: 1,
            b: 2,
            fn: function () {
                var  c = 90;//局部变量
                console.log(this);
                console.log(this.a + this.b);
            }
        };
        /**
            1，谁在调用这个函数，this就指向谁；
            2，这里是obj在调用函数fn,所以里面的this指向obj;
        */
        obj.fn();
        /* 这里函数并没有执行，只是拿出来放到变量fn里面*/
        var f123 = obj.fn;
        f123();// window.f123();
        window.f123();

        obj.f123();
    </script>
```

嵌套函数：

在函数中声明的函数就是嵌套函数，嵌套函数只能在当前函数内访问，好好说外无法，访问

```
function fn () {
    function foo(){
        console.log('woshiqiantaohanshu');
    }
    foo();
}
fn();
```

匿名函数：

没有名字的函数就是匿名函数，它可以让一个变量来接受，也就是用“函数表达式”方式创建和接收

```
var fun = function () {
    alert("我是一个匿名函数");
}

fun();
```

立即执行函数：

函数定义完，立即被调用，这种函数叫做立即执行函数，立即执行函数往往只会执行一次。

```
(function () {
    alert("我是一个匿名函数");
})();
```